(4)

CHEMICAL
RESEARCH,
DEVELOPMENT &
ENGINEERING
CENTER

CRDEC-TR-88091



# STICK: A MOLECULAR STRUCTURE DISPLAY PROGRAM

by Joseph M. Leonard, Ph.D.

RESEARCH DIRECTORATE

May 1988



U.S. ARMY
ARMAMENT
MUNITIONS
CHEMICAL COMMAND

Aberdeen Proving Ground, Maryland 21010-5423

## Disclaimer

The findings in this report are not to be construed as an official Department of the Army position unless so designated by other authorizing documents.

## Distribution Statement

Approved for public release; distribution is unlimited.

PAGES _____

ARE

MISSING

IN

ORIGINAL

DOCUMENT

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE

# REPORT DOCUMENTATION PAGE

| | |
|---|---|
| 1a. REPORT SECURITY CLASSIFICATION<br>UNCLASSIFIED | 1b. RESTRICTIVE MARKINGS |
| 2a. SECURITY CLASSIFICATION AUTHORITY | 3 DISTRIBUTION / AVAILABILITY OF REPORT<br>Approved for public release; distribution is unlimited. |
| 2b. DECLASSIFICATION / DOWNGRADING SCHEDULE | |
| 4. PERFORMING ORGANIZATION REPORT NUMBER(S)<br>CRDEC-TR-88091 | 5. MONITORING ORGANIZATION REPORT NUMBER(S) |
| 6a. NAME OF PERFORMING ORGANIZATION<br>CRDEC<br>6b OFFICE SYMBOL (If applicable)<br>SMCCR-RSP-C | 7a. NAME OF MONITORING ORGANIZATION |
| 6c. ADDRESS (City, State, and ZIP Code)<br>Aberdeen Proving Ground, MD 21010-5423 | 7b. ADDRESS (City, State, and ZIP Code) |
| 8a. NAME OF FUNDING / SPONSORING ORGANIZATION<br>CRDEC<br>8b. OFFICE SYMBOL (If applicable)<br>SMCCR-RS | 9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER |
| 8c. ADDRESS (City, State, and ZIP Code)<br>Aberdeen Proving Ground, MD 21010-5423 | 10 SOURCE OF FUNDING NUMBERS<br>PROGRAM ELEMENT NO. / PROJECT NO. 1C162706 / TASK NO. A553A / WORK UNIT ACCESSION NO. |

11. TITLE (Include Security Classification)
STICK: A Molecular Structure Display Program

12. PERSONAL AUTHOR(S)
Leonard, Joseph M., Ph.D.

| 13a. TYPE OF REPORT | 13b. TIME COVERED | 14. DATE OF REPORT (Year, Month, Day) | 15. PAGE COUNT |
|---|---|---|---|
| Technical | FROM 87 Mar TO 87 Apr | 1988 May | 50 |

16. SUPPLEMENTARY NOTATION

| 17. COSATI CODES | | | 18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number) |
|---|---|---|---|
| FIELD | GROUP | SUB-GROUP | |
| 13 | 2 | | Molecular Modeling, Computational Chemistry, Molecular Graphics |
| 7 | 4 | | |

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

Molecular Graphics provide a means of representing the tremendous amount of information generated by theoretical or computational models in a form that is readily understandable. The Molecular Structure Display program STICK provides the chemist with static, two-dimensional monochrome or color displays of molecular structure using an inexpensive graphics terminal or microcomputer. Currently, STICK supports the Tektronix 4010 and 4105 family of terminals, along with terminals (or software) that emulate either of these Tektronix terminals.

| | |
|---|---|
| 20. DISTRIBUTION / AVAILABILITY OF ABSTRACT<br>☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT. ☐ DTIC USERS | 21. ABSTRACT SECURITY CLASSIFICATION<br>UNCLASSIFIED |
| 22a. NAME OF RESPONSIBLE INDIVIDUAL<br>SANDRA J. JOHNSON | 22b. TELEPHONE (Include Area Code)<br>(301) 671-2914<br>22c. OFFICE SYMBOL<br>SMCCR-SPS-T |

DD FORM 1473, 84 MAR 83 APR edition may be used until exhausted. All other editions are obsolete.

SECURITY CLASSIFICATION OF THIS PAGE
UNCLASSIFIED

CONTENTS

Blank

# STICK: A MOLECULAR STRUCTURE DISPLAY PROGRAM

## 1. INTRODUCTION

Computational Chemistry software systems, such as the Molecular Modeling, Analysis and Display System (MMADS),* provide a "Chemist's Workbench" - a user-friendly, integrated software system organized around a common data structure. Via MMADS, a chemist can generate and visualize two- and three-dimensional molecular structures; optimize structures using molecular mechanics, semiempirical or ab initio quantum chemical techniques; generate physical, physicochemical, or electronic parameters; and investigate the relationship(s) among members of a family of molecules.

MMADS provides a wide range of molecular graphics capabilities, ranging from simple molecular connectivity descriptions to three dimensional, color-coded, electronic surfaces. The software package STICK lies toward the simpler end of the spectrum, providing the chemist with static, two dimensional monochrome, or color molecular displays on an inexpensive graphics terminal (or microcomputer).

STICK provides many capabilities. Structures can be viewed as a stick figure, "ball-and-stick" representation or contact-sphere representation. Color coding is used to describe the atom pair for each bond, as well as each atom, in the "ball-and-stick" and contact-sphere representations. Both orthographic and depth cued projections can be generated. Atoms can be labeled by atom type or formal charge.** Individual atom types can be omitted from the display. Up to ten structures can be simultaneously displayed. Structures can be rotated about or translated along any of the Cartesian axes. Structures can be saved to disk or scaled to a common coordinate system. Bond Stretch/Bend and Torsional strain energies (as calculated by MM2) can be displayed by using color coding to represent absolute magnitude.

---

*Joseph M. Leonard, A User's Guide to the Molecular Modeling, Analysis and Display System (MMADS), CRDEC-TR-86039, U.S. Army Chemical Research Development and Engineering Center, Aberdeen Proving Ground, MD, May 1986.

**This option assumes a formal charge file has been created prior to entering STICK.

Currently, STICK supports the Tektronix 4010 and 4105 family of terminals along with terminals (or software) that emulate either of the Tektronix terminals. Future development will include support for the Adage 3000 raster graphics system, the Digital Equipment Corporation uVAX-II/GPX and the Silicon Graphics IRIS Color Workstations. While this version of STICK is limited to static, two-dimensional displays, codes developed for the color workstations (and the Adage) will provide additional capabilities (e.g., real-time animation).

## 2. STICK COMMANDS

STICK commands can be broken down into four categories: plot format, plot content, plot orientation, and general-purpose. Plot format commands allow the user to define the representation, select the color coding scheme, and indicate what, if any, atoms will be omitted from the display. Plot content commands allow the user to load, save and display multiple structures, and specify the labeling convention to be used in generating the display. Plot orientation commands allow the user to rotate or translate individual structures (or the entire display), and change the scaling of the display. General-purpose commands allow the user to control the operation of STICK. The individual commands are described below with verticals separating valid operands (if any) of each command.

### 2.1 Plot Format.

STICK — This command instructs STICK to produce a "stick figure" representation for all structures loaded into the display. All bonds are indicated and color-coded by the atom pair making the bond. This is the default representation.

BSTICK — This command instructs STICK to produce a "ball and stick" representation for all structures loaded into the display. All bonds are indicated, and all atoms are color-coded by type.

SOLID — This command instructs STICK to produce a contact-sphere representation for all structures loaded into the display. All bonds are indicated, and all atoms are color-coded by type. In this representation, the "perfect" bond is represented by two circles tangent at one point.

STYLE STICKS | BOXES | <null>

This command instructs STICK to display all bonds as line segments (STICKS) or to represent all bonds as polygons scaled to provide depth cueing (BOXES). If STYLE is entered without an operand, the current style is provided. The default style is STICKS.

OPTION COLOR | WHITE | NONE | <null>

This command instructs STICK to color-code all bonds to reflect the atom pair (COLOR), color all bonds white (WHITE), or omit all bonds from the display (NONE). If OPTION is entered without an operand, the current option is provided. The default option is COLOR.

OMIT <atom1> ... <atomn> | <null>

This command instructs STICK to omit the specified atom types from the display. The OMIT command accepts multiple operands, and STICK accumulates the atom types for omission if several OMIT commands are issued. If OMIT is entered without an operand, the atom types that will be omitted are provided. Unless otherwise specified, all atoms are displayed.

STRAIN STRETCH | BEND | TORSIONAL | <null>

This command instructs STICK to color-code all bonds to reflect the bond stretch, bend, or torsional energy associated with each internal coordinate of the molecule. This command requires information obtained from the MM2 answer file, and the user is informed if the data is unavailable. A spectral color-coding scheme is used to indicate energies in the range [0.04,4.0+] Kcal. If STRAIN is entered without an operand, the current option is provided.

2.2 Plot Content.

LOAD <filename>

This command loads the specified structure file into STICK using the coordinates specified in the input file. If appropriate, the display is rescaled to contain the new structure. If more than one file has been loaded, STICK omits the title line of the display.

**The current structure (defined when STICK is entered) is structure #1, and STICK permits a maximum of 10 structures to be LOADed.**

**SAVE** **<number>,<filename> | <filename>**

**This command saves structure <number> in the specified structure file using the current coordinate system of the display. If <number> is omitted, all structures are saved in the specified file. This option has turned out to be a useful side effect of STICK and provides a graphical means of merging chemical structures in a common coordinate system.**

USE <number> | ALL | <null>

This command specifies that structure <number> is to be used by the rotation and translation commands, or that the entire display will be used. If USE is entered without an operand, the name and number of all structure files that have been LOADed are provided. Unless otherwise specified, structure #1 is USEd.

VIEW <number> | <null>

This command toggles whether structure <number> will be visible or invisible on the display. If VIEW is entered without an operand, the view status of all structures is provided. Unless otherwise specified, all structures are visible.

LABEL LABELS | FORMAL | <null>

This command instructs STICK to label each atom in the display by its type and number (LABELS) or by its formal charge (FORMAL). On color terminals, the formal charge is color-coded by sign: magenta for positive values and red for negative values. For STICK and BSTICK representations, STICK endeavors to place the labels so they are not obscured by atoms in the display. For SOLID representations, the labels are placed at the center of each atom in the display.

THRESHOLD <value> | <null>

This command instructs STICK to omit the formal charge label for any atom with |FC| < <value>. Due to

limitations of the graphics hardware, 0.01 is enforced as the minimum threshold value. If THRESHOLD is entered without an operand, the current value is provided.

## 2.3 Plot Orientation.

XROT <value>

This command instructs STICK to rotate the current structure (defined by the USE command) by <value> degrees about the x-axis.

YROT <value>

This command instructs STICK to rotate the current structure (defined by the USE command) by <value> degrees about the y-axis.

ZROT <value>

This command instructs STICK to rotate the current structure (defined by the USE command) by <value> degrees about the z-axis.

XTRANS <value>

This command instructs STICK to translate the current structure (defined by the USE command) by <value> Angstroms along the x-axis. This command has no observable effect if the entire display is acted upon.

YTRANS <value>

This command instructs STICK to translate the current structure (defined by the USE command) by <value> Angstroms along the y-axis. This command has no observable effect if the entire display is acted upon.

ZTRANS <value>

This command instructs STICK to translate the current structure (defined by the USE command) by <value> Angstroms along the z-axis. This command has no observable effect if the entire display is acted upon.

SCALE + | -

This command instructs STICK to increase (+) the size of the display by 10% or decrease (-) the size of the display by 10%. Any alterations made via the SCALE command will be lost when rescaling is applied after a rotation or translation command is performed.

BLACK This command specifies that a dark background will be used in subsequent plots.

GREY This command specifies that a light background will be used in subsequent plots.

2.4 General Purpose.

PLOT This command instructs STICK to generate a new display using the various display parameters described in the previous commands.

RESET This command reinitializes STICK as if the user exited and reentered the program. All parameters are restored to their default values, and all LOADed structures are deleted.

EXIT This command exits STICK.

## 3. CONCLUSION

STICK provides several useful capabilities to the computational chemist in a single, easy-to-use package. Using STICK, a molecule can be viewed in several representations. The electronic environment (via the formal charges) can be sampled at each atomic site, and the molecular strain energy (generated via molecular mechanics) can be evaluated for each internal coordinate.

STICK's ability to display several molecules simultaneously provides a visually effective means of comparing a family of molecules. This feature also permits a researcher to orient several molecular fragments in relation to each other and save the result for later work. Finally, STICK has been used to prepare data files for submission to semiempirical quantum chemical programs in which a molecular ion is surrounded by a varying number of water molecules.

# APPENDIX A

# MMADS GENERIC GRAPHICS LIBRARY

Blank

## APPENDIX A

## MMADS GENERIC GRAPHICS LIBRARY

The MMADS Generic Graphics Library resulted from the need to create a small, easily modified, device-independent graphics library. Because all machine-dependent codes would be contained in the graphics library (transparent to the programmer), such a library would enable MMADS developers to maintain a single version of code for each of the graphics commands in MMADS.

Currently, the Tektronix 4010 and 4105 terminals, along with the hardware that emulates either of these devices, are supported. Future development will include support for the Adage 3000 raster graphics system and the Digital Equipment Corporation uVAX-II/GPX and Silicon Graphics IRIS 3120 Color Workstations. Obviously, when compared to the Tektronix terminals, the color workstations greatly enhance hardware capabilities. However, all future versions of the MMADS Generic Graphics Library will remain compatible with this implementation.

Available Commands

clear_dialog — Clear Dialog (Text) Screen

call: call clear_dialog

calling parameters: none

This subroutine clears the dialog (text) area of the display.

clear_grafix — Clear Graphics Screen

call: call clear_grafix

calling parameters: none

This subroutine clears the graphics area of the display.

draw_circle  Draw a Circle

call: call draw_circle(x,y,radius,number, fill,border)

calling parameters:

| | |
|---|---|
| x | origin x-coordinate (internal coordinates) |
| y | origin y-coordinate (internal coordinates) |
| radius | circular radius (internal coordinates) |
| number | number of line segments used to draw circle |
| fill | fill pattern to use (if available) |
| border | = yes_value, include the border<br># yes_value, exclude the border |

This subroutine draws a circle centered at (x,y), with a radius of <radius>, using <number> line segments. The value of <fill> specifies whether the circle will be empty or filled with the specified fill pattern. The value of <border> specifies whether the circle will be drawn with a border.

draw_poly  Draw a Polygon

call: call draw_poly(x,y,number,fill,border)

calling parameters:

| | |
|---|---|
| x | origin x-coordinate (internal coordinates) |
| y | origin y-coordinate (internal coordinates) |
| number | number of points defining the polygon |
| fill | fill pattern to use (if available) |
| border | = yes_value, include the border<br># yes_value, exclude the border |

This subroutine draws a polygon with <number> vertices starting at [x(1),y(1)] and ending at [x(number),y(number)]. The value of <fill> specifies whether the polygon will be empty or filled with the specified fill pattern. The value of <border> specifies whether the polygon will be drawn with a border.

draw_to — Draw a Line

call: call draw_to(x,y)

calling parameters:

x point x-coordinate (internal coordinates)
y point y-coordinate (internal coordinates)

This subroutine draws a line from the current cursor position to (x,y).

enter_grafix — Enable Graphics Device

call: call enter_grafix(lun,device)

calling parameters:

lun graphics device logical unit number
device = t4105, graphics device is a Tek 4105
= t4010, graphics device is a Tek 4010

This subroutine initializes the MMADS Generic Graphics Library. It is assumed that the calling routine has ensured that the graphics device is linked to <lun>, which will be used for all graphics input/output (i/o). <device> specifies the graphics device type.

exit_grafix — Disable Graphics Device

call: call exit_grafix

calling parameters: none

This subroutine disables the MMADS Generic Graphics Library and resets the graphics device to ANSI mode.

flush — Flush Graphics I/O Buffer

call: call flush

calling parameters: none

This subroutine flushes the i/o buffer used by the MMADS Generic Graphics Library. This should be

done after the calling program has completed a plot to ensure that all of the information is sent to the display.

frame — Frame Graphics Display

call: call frame(xmin,xmax,ymin,ymax)

calling parameters:

xmin minimum internal x-coordinate
xmax maximum internal x-coordinate
ymin minimum internal y-coordinate
ymax maximum internal y-coordinate

This subroutine sets the x and y coordinate ranges that will be used to map internal coordinates to the hardware pixel addresses.

line_color — Set Graphics Line Color

call: call line_color(color)

calling parameter:

color desired line color [integer in the range (0,7)]

This subroutine sets the line color to be used by draw_to, draw_circle, and draw_poly. The programmer is referred to the appropriate hardware manual for a description of the available color codes.

line_style — Set Graphics Line Style

call: call line_style(style)

calling parameter:

style desired line style

This subroutine sets the line style to be used by draw_to. Currently, this routine supports only solid (style = 0) and dotted (style = 4) lines.

move_to — Move the Cursor

call: call move_to(x,y)

calling parameters:

x point x-coordinate (internal coordinates)
y point y-coordinate (internal coordinates)

This subroutine sets the current cursor position to (x,y).

output_text — Write Text on the Graphics Screen

call: call output_text(x,y,string,length)

calling parameters:

x lower-left x-coordinate of text (internal coordinates)
y lower-left y-coordinate of text (internal coordinates)
string text to be written
length number of characters in <string>

This subroutine writes the specified text string on the graphics screen at (x,y).

scale_x — Map Internal X-Coordinate to a Pixel Address

call: call scale_x(x)

calling parameter:

x internal x-coordinate

This integer function returns the hardware pixel address corresponding to <x> using the internal coordinate system defined via the FRAME command. Values of <x> outside the range (xmin,xmax) will be set to the appropriate extremum value. scale_x assumes that the display area is a square and performs the mapping function accordingly (using the pixel address range of the graphics device).

scale_y — Map Internal Y-Coordinate to a Pixel Address

call: call scale_y(y)

calling parameter:

y internal y-coordinate

This integer function returns the hardware pixel address corresponding to <y> using the internal coordinate system defined via the FRAME command. Values of <y> outside the range (ymin,ymax) will be set to the appropriate extremum value. scale_y assumes that the display area is a square and performs the mapping function accordingly (using the pixel address range of the graphics device).

surface_color — Set an Entry in the Color Map

call: call surface_color(index,hue,light,satur)

calling parameters:

index the color index to set
hue the hue index (HLS coding)
light the lightness index (HLS coding)
satur the saturation index (HLS coding)

This subroutine sets the HLS indices for the specified color index. This enables the programmer to alter the default color settings of the graphics device based on the specific requirements of the graphics application. It is strongly suggested that the programmer restore all color indices to their default setting before exiting.

text_color — Set Graphicstext Color

call: call text_color(color)

calling parameter:

color desired text color [integer in the range (0,7)]

This subroutine sets the color that will be used for text written by output_text. As with surface_color, it is strongly suggested that the programmer restore the text color to the default setting (white) before exiting.

write_mode Set Graphicstext Writing Mode

call: call write_mode(mode)

calling parameters:

mode = 1, replace existing display
= 0, overstrike existing display

This subroutine specifies whether text written by output_text replaces or overstrikes the existing display. As with surface_color, it is strongly suggested that the programmer set the graphics terminal to the default mode (overstrike) before exiting.

Blank

# APPENDIX B

# MMADS GRAPHICS PRIMITIVE LIBRARY

Blank

## APPENDIX B

## MMADS Graphics Primitive Library

The MMADS Graphics Primitive Library contains the subroutines that perform the actual hardware operations on the graphics device. As such, these routines can be called from the graphics application directly; but, it is strongly suggested that the MMADS Generic Graphics Library be used as an interface. The Tektronix (TEK) 4010 and 4105 terminals are the two graphics devices currently supported.

The Primitive Library was designed with simplicity in mind. Each subroutine contains the code necessary to perform only a single graphics operation. The subroutine name can be broken down into the device type, TEK (TEK 4105) or P10 (TEK 4010), and the mnemonic for the operation (e.g., SLI for Set Line Index). As additional graphics devices are added, this naming convention will be expanded to segregate the primitives.

To improve the performance of the graphics application, all input/output (i/o) requests are buffered and sent to the graphics device via large, unformatted write operations. This greatly reduces both the operating system and Fortran Format processing overhead. This feature is only implemented for the Tektronix 4105 terminal.

Available Primitives (Tek 4105)

tekbpb Begin Panel Boundary

call: call tekbpb(ix,iy,incl)

calling parameters:

ix pixel x-coordinate
iy pixel y-coordinate
incl = 1, include the panel boundary
= 0, exclude the panel boundary

This routine marks the start of a panel on the graphics screen and specifies whether the panel will be drawn with a boundary.

tekcrd Generate a Coordinate String

call: call tekcrd(ix,iy,array)

calling parameters:

ix pixel x-coordinate
iy pixel y-coordinate
array bit string to send to the terminal

This subroutine constructs the five-byte bit string required to specify a pixel address on the graphics screen.

tekdcl Clear Dialog Screen

call: call tekdcl

calling parameters: none

This subroutine clears the dialog area of the display.

tekdrw Draw a Line

call: call tekdrw(ix,iy)

calling parameters:

ix pixel x-coordinate
iy pixel y-coordinate

This subroutine draws a line from the current cursor position to (ix,iy).

tekepn End a Panel Definition

call: call tekepn

calling parameters: none

This subroutine indicates that the current cursor position is the last point in a panel, forcing a line connecting this point with the point specified by tekbpb. Any panel defined via tekbpb/tekepn will be a closed polygon filled with the current fill pattern.

tekfls Flush Graphics Buffer

call: call tekfls

calling parameters: none

This subroutine flushes the graphics buffer and should be called at the end of each plot to ensure that all information has been sent to the graphics screen.

tekgcl Clear Graphics Screen

call: call tekgcl

calling parameters: none

This subroutine clears the graphics area of the display.

tekint Generate an Integer String

call: call tekint(integ,tint,ntint)

calling parameters:

integ integer to be encoded
tint bit string to send to the terminal (in reverse order)
ntint number of characters in <tint>

This subroutine constructs the variable-length bit string required to encode an integer to send to the graphics terminal.

tekmod Set Graphics Terminal Mode

call: call tekmod(mode)

calling parameters:

mode = 0, set TEK mode (graphics)
= 1, set ANSI mode (VT100)

This subroutine sets the operating mode for the graphics terminal to TEK (graphics) or ANSI (VT100) mode. The programmer is referred to the appropriate

hardware manual for a description of the commands available in each mode.

tekmov  Move the Cursor

call:  call tekmov(ix,iy)

calling parameters:

ix  pixel x-coordinate
iy  pixel y-coordinate

This subroutine sets the current cursor position to (ix,iy).

tekout  Perform Buffered I/O to the Graphics Device

call:  call tekout(length,string)

calling parameters:

length  the number of bytes in <string>
string  the escape sequence to send to the terminal

This subroutine concatenates all TEK 4105 escape sequences, sending output to the graphics lun (assumed to be open) in large, unformatted blocks. This greatly reduces the i/o overhead incurred by the graphics application.

tekrst  Reset the Color Settings

call:  call tekrst

calling parameters:  none

This subroutine resets the hardware color wheel to the factory default settings and sets the line color for the graphics screen to white. The programmer is referred to the appropriate hardware manual for a description of the default colors.

tekscm  Set Surface Color Map Entry

call:  call tekscm(index,hue,light,satur)

calling parameters:

index  color index to be redefined
hue    HLS hue index (integer)
light  HLS lightness index (integer)
satur  HLS saturation index (integer)

This subroutine resets the HLS indices associated with the specified color index.

teksfp  Select Fill Pattern

call:  call teksfp(icolor)

calling parameter:

icolor  color index or hardware dither pattern index

This subroutine sets the fill pattern (solid color or dither pattern) to be used when drawing panels. The programmer is referred to the appropriate manual for a description of the available dither patterns.

teksgawm  Set Graphics Area Writing Mode

call:  call teksgawm(mode)

calling parameters:

mode  = 0, replace existing display
      = 1, overstrike existing display

This subroutine specifies whether text written on the graphics screen replaces or overstrikes the existing display.

teksli  Set Line Index

call:  call teksli(color)

calling parameter:

color  color index to use (integer)

This subroutine sets the line color to be used on the graphics screen.

teksls Set Line Style

call: call teksls(code)

calling parameters:

code = 0, use solid lines (-----)

= 4, use dotted lines (- - -)

This subroutine sets the line style to be used on the graphics screen. The programmer is referred to the appropriate manual for a description of the available line styles.

teksti Set Text Index

call: call teksti(color)

calling parameter:

color color index to use (integer)

This subroutine sets the color for text written on the graphics screen.

tekwrt Write Text on the Graphics Screen

call: call tekwrt(ix,iy,string,length)

calling parameters:

| | |
|---|---|
| ix | pixel x-coordinate |
| iy | pixel y-coordinate |
| string | text to be written |
| length | number of characters in <string> |

This subroutine writes the specified text string on the graphics screen at (ix,iy).

Available Primitives (TEK 4010)

p10crd Generate a Coordinate String

call: call p10crd(ix,iy,array)

calling parameters:

ix pixel x-coordinate
iy pixel y-coordinate
array bit string to send to the terminal

This subroutine constructs the four-byte bit string required to specify a pixel address on the graphics screen.

p10dcl Clear Dialog Screen

call: call p10dcl

calling parameters: none

This subroutine clears the dialog area of the display.

p10drw Draw a line

call: call p10drw(ix,iy)

calling parameters:

ix pixel x-coordinate
iy pixel y-coordinate

This subroutine draws a line from the current cursor position to (ix,iy).

p10gcl Clear Graphics Screen

call: call p10gcl

calling parameters: none

This subroutine clears the graphics area of the display.

p10mod Set Graphics Terminal Mode

call: call p10mod(mode)

calling parameters:

mode = 0, set TEK mode (graphics)
= 1, set ANSI mode (VT100)

This subroutine sets the operating mode for the graphics terminal to TEK (graphics) or ANSI (VT100) mode. The programmer is referred to the appropriate hardware manual for a description of the commands available in each mode.

p10mov Move the Cursor

call: call p10mov(ix,iy)

calling parameters:

ix pixel x-coordinate
iy pixel y-coordinate

This subroutine sets the current cursor position to (ix,iy).

p10sls Set Line Style

call: call p10sls(code)

calling parameters:

code = 0, use solid lines (-----)
= 4, use dotted lines (- - -)

This subroutine sets the line style to be used on the graphics screen. Due to compatibility restrictions, the style codes indicated above are translated to the actual hardware values. The programmer is referred to the appropriate manual for a description of the available line styles.

p10wrt Write Text on the Graphics Screen

call: call p10wrt(ix,iy,string,length)

calling parameters:

ix pixel x-coordinate
iy pixel y-coordinate
string text to be written
length number of characters in <string>

This subroutine writes the specified text string on the graphics screen at (ix,iy).

# APPENDIX C

# MMADS INPUT PARSER

Blank

## APPENDIX C

## MMADS INPUT PARSER

The MMADS Input Parser consists of three parts: (1) a routine to get a token from an input string (either space or comma terminated); (2) a routine to compare a token against a list of commands; and (3) a routine to decode a token to a real number or a collection of integers.

These three routines are used by STICK (and throughout the MMADS code that has been written at CRDEC) to provide a consistent and straightforward user-interface. Commands can be entered in either upper or lower case, and abbreviations are acceptable. The user is informed if an abbreviated command is ambiguous.

### Parsing Routines

parse  Get a Token from a String

call: ierr=parse(string,length,token,len2)

calling parameters:

| | | |
|---|---|---|
| string | input string, returned with token removed | |
| length | length of <string>, returned with token removed | |
| token | token removed from <string> | |
| len2 | length of <token> | |
| ierr | termination code: | = 0, normal return |
| | | = 1, EOL terminated |
| | | = 2, null token (len2 = 0) |

This function accepts a string of length <length> from the caller and scans for the first occurrence of a space or a comma. If either are found, the characters preceding the separator are returned as <token> with a length of <len2> and removed from <string> (with <length> adjusted accordingly). The return code indicates whether <token> was found and whether a separator or EOL was encountered.

parsec  Check Token Against Command List

call: ierr=parsec(string,length,table,entry)

calling parameters:

| | | |
|---|---|---|
| string | the token to be checked | |
| length | length of <string> | |
| table | command table, terminated by '*END*' | |
| entry | explicit command, returned to caller | |
| ierr | termination code: | = 0, normal termination |
| | | = 1, invalid command |
| | | = 2, ambiguous command |

This function accepts a string of length <length> from the caller and compares it against the first <length> bytes of the character strings contained in <table>. The number of matches determine whether the command is valid, ambiguous, or invalid. If <string> is a valid command, parsec returns the fully-specified command in <entry>.

parsen Decode Token to Real/Integers

call: ierr=parsen(type,string,length,array,numb1, iarray,numb2,maxnum)

calling parameters:

| | | |
|---|---|---|
| type | = 0, real number | |
| | = 1, integer number(s) | |
| string | character string to process | |
| length | length of <string> | |
| array | array containing real numbers | |
| numb1 | number of real numbers in <array> (currently, 1) | |
| iarray | array containing integers | |
| numb2 | number of integers in <iarray> | |
| maxnum | for integers, largest possible number | |
| ierr | termination code: | = 0, normal return |
| | | = 2, invalid number string |

This function accepts a character string of length <length> and processes it according to the value of <type> passed by the caller. If integers are desired, "-" is treated as a dash [as in 5-10 (5 through 10)]. Negative numbers are not permitted, and the maximum valid integer is <maxnum>. If a real number is desired, "-" is a minus sign, and a decimal point (if omitted) will be assumed to follow the last digit in <string>.

# APPENDIX D

## COLOR AND RADIUS TABLE USED BY STICK

Blank

APPENDIX D

COLOR AND RADIUS TABLE USED BY STICK

| Atom | Radius (A)* | Solid Color** | Fill Pattern** |
|---|---|---|---|
| H | 0.32 | 1 | 110 |
| O | 0.73 | 2 | -2 |
| N | 0.75 | 5 | -5 |
| C | 0.77 | 4 | 9 |
| P | 1.06 | 6 | -6 |
| S | 1.02 | 7 | -7 |
| F | 0.72 | 3 | -3 |
| Cl | 0.99 | 3 | -3 |
| Br | 1.14 | 3 | -3 |
| I | 1.33 | 3 | -3 |
| Si | 1.11 | 4 | -4 |

*Radii reduced by 2.5X for "ball-and-stick" representation
**Solid color and fill patterns for the TEK 4105

Blank

# APPENDIX E

## VAX/VMS BUILD FILE AND MMADS COMMAND FILE

Blank

## APPENDIX E

## VAX/VMS BUILD FILE AND MMADS COMMAND FILE

Build File for STICK

```
$ fortran STICK2,GRAFIX,GRAFIX_PRIM,PARSE
$ link    STICK2,GRAFIX,GRAFIX_PRIM,PARSE
$ purge   *.FOR,STICK2.EXE
```

MMADS Command File for STICK

```
$!
$! STICK -- Stick figure display on the terminal
$!
$! Modified 1/29/87 for the uVAX-II (jle)
$!
$! Set the process name
$!
$ @trb$command:setproc STICK
$!
$! Look for the structure file
$!
$ file = trb$structure + ".con"
$ look = f$search(file)
$ if look .eqs. "" then goto nocon
$!
$! Check on the terminal type
$!
$ if trb$tty_type .eqs. "@@@" then @trb$command:deftty
$ if trb$tty_type .eqs. "*" then goto no_vt100
$!
$! Create the marker file
$!
$ open/write output 'trb$structure'.mrk
$ write output trb$tty_type
$ close output
$!
$ assign/user 'trb$structure'.fch for010
$ assign/user 'trb$structure'.mrk for004
$ define/user sys$input sys$command:
$ assign/user tt:  for003
$ assign/user 'trb$structure'.con for001
$ run trb$system:stick2
```

```
$ delete 'trb$structure'.mrk;
$ exit
$!
$!No structure file was found
$!
$ nocon:
$!
$ write sys$output "Unable to locate the structure file"
$ exit
$!
$ no_vt100:
$!
$ write sys$output "STICK does not currently support the VT100"
```

# APPENDIX F

# FILE FORMATS USED BY STICK

Blank

## APPENDIX F

## FILE FORMATS USED BY STICK

### Structure File

Record #1 - the header record for the file

| | | |
|---|---|---|
| Column Nos. 1-3 | (i3) | The number of atoms contained in the file. |
| Column Nos. 4-72 | (a) | The title of the structure file. |

Record #2 on - the descriptions of the individual atoms.

| | | |
|---|---|---|
| Column No. 1 | (1x) | blank |
| Column Nos. 2-3 | (a) | The atom symbol. Single letter labels must be preceded by a space. |
| Column Nos. 4-8 | (i5) | The atom index. |
| Column Nos. 9-20 | (f12.6) | The x-coordinate. |
| Column Nos. 21-32 | (f12.6) | The y-coordinate. |
| Column Nos. 33-44 | (f12.6) | The z-coordinate. |
| Column Nos. 45-49 | (i5) | The MM2 atom type (3).* |
| Column Nos. 50-79 | (6i5) | The bond connectivity. |

### Formal Charge File

Record #1-on:

| | | |
|---|---|---|
| Column Nos. 1-5 | (i5) | The atom index. |
| Column Nos. 6-15 | (f10.6) | The formal charge. |

---

*(3) Atom types used for the MM2 Molecular Mechanics program developed by Allinger et al., 1980.

Bond Strain File

Record #1:

| | | |
|---|---|---|
| Column Nos. 1-5 | (i5) | The number of bonds in the molecule |

Record #2-on:

| | | |
|---|---|---|
| Column Nos. 1-5 | (i5) | The index of the first bound atom |
| Column Nos. 6-10 | (i5) | The index of the second bound atom |
| Column Nos. 11-20 | (f10.6) | The bond strain energy (Kcal) |

Angle Strain File

Record #1:

| | | |
|---|---|---|
| Column Nos. 1-5 | (i5) | The number of angles in the molecule |

Record #2-on:

| | | |
|---|---|---|
| Column Nos. 1-5 | (i5) | The index of the first bound atom |
| Column Nos. 6-10 | (i5) | The index of the second bound atom |
| Column Nos. 11-15 | (i5) | The index of the third bound atom |
| Column Nos. 16-25 | (f10.6) | The angle strain energy (Kcal) |

Torsional Strain File

Record #1:

| | | |
|---|---|---|
| Column Nos. 1-5 | (i5) | The number of torsional angles in the molecule |

Record #2-on:

| | | |
|---|---|---|
| Column Nos. 1-5 | (i5) | The index of the first bound atom |
| Column Nos. 6-10 | (i5) | The index of the second bound atom |

| | | |
|---|---|---|
| Column Nos. 11-15 | (i5) | The index of the third bound atom |
| Column Nos. 16-20 | (i5) | The index of the fourth bound atom |
| Column Nos. 21-30 | (f10.6) | The torsional angle strain energy (Kcal) |

## Example - Structure File

```
6 Methanol - STICK demo
C    1    1.459980    -0.027340    1.314950    1    5    4    3    2
O    2    0.964510    -0.006410   -0.002090    6    6    1
H    3    0.597940     0.059200    2.014660    5    1
H    4    2.147630     0.840190    1.429940    5    1
H    5    2.007030    -0.985220    1.467960    5    1
H    6    0.361040    -0.760410   -0.117010   21    2
```

## Example - Formal Charge File

```
1    0.191646
2   -0.397017
3   -0.012174
4    0.007766
5   -0.012176
6    0.221955
```

## Example - Bond Strain File

```
5 bonds in file
1     2     0.0000
1     3     0.0001
1     4     0.0000
1     5     0.0001
2     6     0.0001
```

## Example - Angle Strain File

```
7 angles in file
2     1     3     0.0311
2     1     4     0.0099
2     1     5     0.0311
3     1     4     0.0193
3     1     5     0.0323
4     1     5     0.0192
1     2     6     0.0199
```

## Example - Torsional Strain File

```
3 torsionals in file
3     1     2     6     0.0000
4     1     2     6     0.0000
5     1     2     6     0.0000
```